Analysis of computed tomography (CT) images

OBSS - 2. Assignment

Nejc Ločičnik
University of Ljubljana
nl4952@student.uni-lj.si

January 14, 2024

Abstract

This work refines the Canny edge detection algorithm for visualizing CT MRI images. Empirical parameter adjustments enhance visualization clarity, balancing detail. The algorithm incorporates edge thinning, 24-connectivity, and additional steps for improved structure representation. Results, presented through animations and 3D point clouds, demonstrate efficacy in medical image visualization, showcasing adaptability to various applications.

1 Introduction

The Canny edge detection algorithm stands as a cornerstone in image processing, particularly valuable in medical imaging. Its application to CT MRI images is vital for highlighting anatomical structures and enhancing diagnostic precision [1]. This study refines the Canny algorithm, tailoring it to the unique demands of CT images, with a focus on improving visual clarity and adaptability. The refined algorithm's efficacy is demonstrated through animations and 3D point cloud visualizations, showcasing its potential contributions to medical image analysis and diagnosis.

2 Methodology

The Canny edge detector is a multi-step algorithm for detecting edges in images (see Figure 1). Its steps include preprocessing, edge detection, edge thinning, and edge linking.

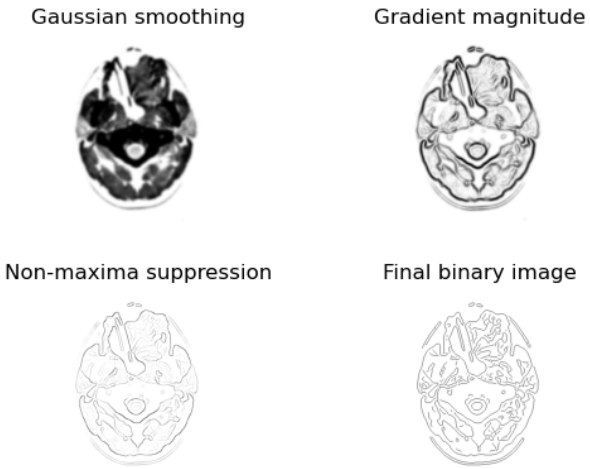

Figure 1: Visualization of the main steps in the Canny edge detector.

2.1 Image Smoothing

The image is initially smoothed with a simple Gaussian filter, described in equation 1. Smoothing is required to reduce noise, preparing the image for edge detection.

$$G(x,y) = \frac{e^{\frac{-x^2+y^2}{2\sigma^2}}}{2\pi\sigma^2} \tag{1}$$

The Gaussian smoothing parameters are empirically determined based on the image size, with $\sigma = \min(\text{width}, \text{height}) \cdot 0.005$, and the kernel size is $6 \cdot \sigma$. Initially, I aimed to fine-tune the parameters to align with the results of Matlab's implementation, but later decided to stick with the empirically

determined values mentioned in the lectures.

2.2 Image gradient

The algorithm proceeds by determining the image gradient using Sobel's operator (kernels G_x and G_y), which is then used to calculate the magnitude (equation 2) and direction (equation 3) of edge pixels.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{2}$$

$$\alpha(x, y) = \arctan(\frac{g_y}{g_x}) \tag{3}$$

2.3 Non-maxima suppression

The obtained edge is then further refined using non-maxima suppression. The edge is thinned in the calculated direction, and only the maximum magnitude is retained, while other pixels are suppressed. This process attempts to narrow the edge to a width of 1 pixel.

2.4 Hysteresis thresholding

Non-maxima suppression is followed by hysteresis thresholding, which differentiates whether an edge pixel is strong or weak. Strong pixels are considered true edges, while weak pixels are transformed into strong ones if they neighbor a true edge. This is a form of edge linking. The result is a binary image where edge pixels are represented by true values, while the rest of the image has false values.

The high threshold is set at 15% of the maximum magnitude value, which is normally 255, while the low threshold is set at 0.5% of the same value. Lowering the thresholds increases the level of detail retained.

2.5 Additional edge thinning

Additionally, an extra step of edge thinning was introduced. Specifically, it thins out diagonal (stair-like) edges (image 2 and 3).

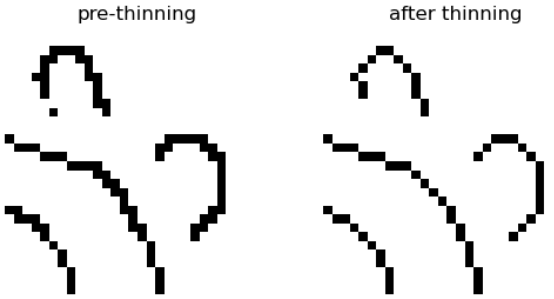

Figure 2: A visual comparison of before and after additional edge thinning.

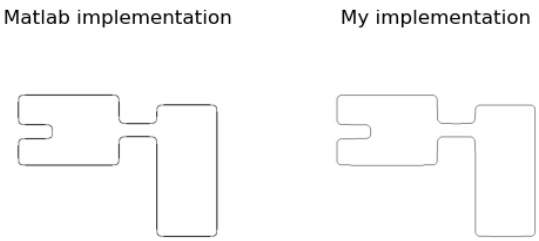

Figure 3: Comparison between Matlab's and my implementation.

2.6 24-Connectivity between neighbouring images

Since we are dealing with a sequence of CT MRI images, we can further link them using 24-connectivity. 24-connectivity adds a pixel to the edge in image n if it would allow it to link with another edge in image n+1. It checks the neighboring 24 pixels, hence the name.

The added pixels aren't very noticeable (Figure 4) if we don't explicitly showcase them. 24-connectivity adds about 1000 pixels per image in the sequence.

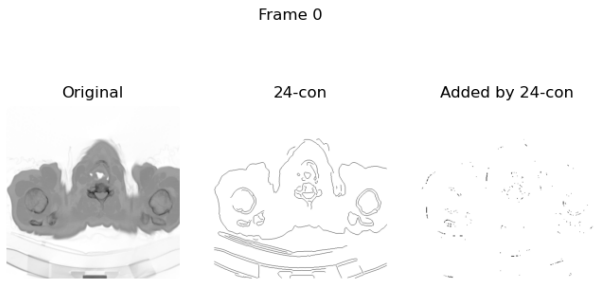

Figure 4: Visualization of 24-connectivity added pixels.

3 Results

Initially, I intended to fine-tune the parameters to match the results of Matlab's implementation. However, after experimenting with 3D visualizations (reconstructions) of the CT images, I opted for parameters that result in less detail. This choice is beneficial for visualization purposes. Specifically, I used a larger kernel and σ for image smoothing (in alignment with what was mentioned during the lectures) and higher thresholds for hysteresis thresholding (also aligned with what was mentioned at the lectures). This approach leads to fewer detected edges (Figure 5), which enhances the point cloud visualization, already challenged by poor visibility.

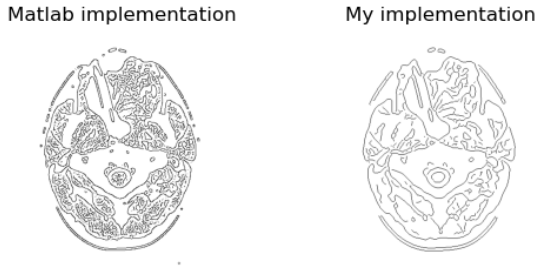

Figure 5: Difference in detected detail.

The results are presented through two different visualizations. The first method involves combining the sequence of images into an animation that navigates through the slices (a snapshot of the animation is shown in Figure 4). The second approach visualizes all the edges in a 3D point cloud (Figure 6). The point cloud visualization is particularly interesting as it allows for clear identification of the spine, ribs, shoulder blades, and various internal organs (although I am not qualified to make specific identifications).

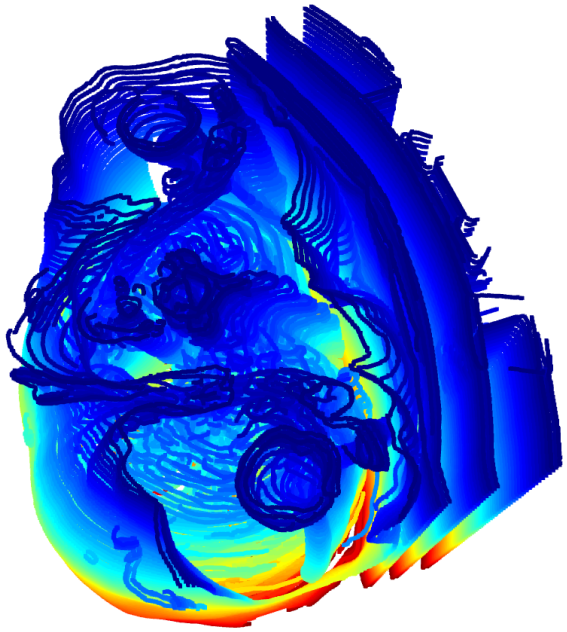

Figure 6: Point cloud visualization with open3D.

4 Discussion

I believe empirically determined parameters are often insufficient as they should be determined on a case-by-case basis, depending on the specific use case for the resulting images. In my situation, given the lack of visual clarity in the point cloud visualization, it made sense to employ parameters resulting in lower details.

The additional step I introduced to the algorithm is edge thinning. I find this step interesting in its implementation, as it can be utilized to preserve and filter for specific patterns in neighboring pixels. In this instance, I employed it to thin stair-like edges.

References

[1] Ctmri images. `https://lbcsi.fri.uni-lj.si/OBSS/Data/CTMRI/`. Accessed: January 14, 2024.